Remarks

Status of application

Claims 1-30 were examined and stand rejected in view of prior art, as well as stand rejected for technical issues. The claims have been amended to further clarify Applicant's invention and to address objections raised by the Examiner. Reexamination and reconsideration are respectfully requested.

The invention

The present invention provides for introducing a cache view of a database for read-only transactions which comprises a separate view of the database at a particular point in time. Transactional log file(s) are used to reconstruct a view of the database by applying log records to this view in the cache, so as to create a version of the database at the particular point in time. Once this cache view is created, for the duration of a given read-only transaction, these blocks cannot be written back to the database (because the write would corrupt them as it is a different view from a different version). However, for implementing the present invention it is desirable that these blocks are not lost. Accordingly, the present invention utilizes a "shadow cache", which itself is implemented via a temporary database table. If the cache is overflowing then the shadow cache is employed to hold the blocks from this old version that have been created (e.g., for use in a read-only transaction).

In the currently preferred embodiment, the shadow cache is implemented by taking advantage of a temporary database that is used in conjunction with a SQL query engine to create a simple shadow table. The shadow table that is created is a very simple shadow table which contains a mapping of the block from the read-only cache view into a block that is to be used as a backing store in the shadow table. The shadow table is a very narrow table and it can be accessed very efficiently. In the presently preferred embodiment of the present invention the shadow table is keyed on block number and it also records a secondary block, which is the block that it maps to a block allocated in the temporary database.

It is important to note that this shadow cache is only used if needed. If the cache view itself does not live long enough to ever need to write any of its blocks to the shadow

cache, then those cache blocks will not be written to the shadow cache and instead will just be invalidated. Applicant's approach is very efficient as unless a transaction has to modify a lot of blocks or lasts a very long time, the shadow cache will likely not be used. The shadow cache is also used to preserve logical undo by preserving the effects of the logical undo operation for the lifetime of a read-only transaction. The shadow cache enables the effects of the logical undo to be saved, which in the currently preferred embodiment allows the logical undo operation to be preserved up front (i.e., at the start of the read-only transaction). Another benefit of the shadow cache is that it enhances system performance.

General

A. Requirement for Information

Applicant has provided copies of the items of information requested by the Examiner in an attachment to this Amendment.

B. Drawings

The Examiner has objected to the drawings under 37 CFR 1.83(a) stating that the drawings must show every feature of the invention specified in the claims. Applicant has amended independent claim 17 and dependent claims thereof to refer to features that are shown in the drawings, thereby overcoming the objection.

C. Amendment to Specification

The Examiner objected to the Abstract as exceeding the limit of 150 words. Applicant has amended the Abstract so that it does not exceed 150 words, thereby overcoming this objection.

The Examiner has also objected to the title of the invention as not being descriptive of the invention to which the claims are directed. Applicant has amended the title of the invention, thereby overcoming this objection.

In accordance with the request of the Examiner, Applicant has amended the specification to include trademark markings for various product names and technology, including Btrieve, Java, and Unix. However, Applicant takes no position as to the

trademark validity or ownership of any such marks of third parties.

The Examiner objected to the disclosure as containing embedded hyperlinks and specifically indicated embedded hyperlinks were found in paragraphs [0027], [0031], and [0054]. Applicant has amended paragraph [0054] to address the Examiner's objection. However, neither paragraph [0027] nor paragraph [0031] includes any URLs which are preceded by "http://" or are between the symbols "< >". As MPEP § 608.01 provides that hyperlinks or a browser-executable code are a URL placed between the symbols "< >" or "http://" followed by a URL address, these paragraphs do not include hyperlinks or browser-executable code. Thus, Applicant respectfully believes the Examiner's rejection is improper as to paragraphs [0027] and [0031] and should be withdrawn.

The Examiner has also objected to disclosure as confusing and inconsistent stating that the terms "shadow cache" and "shadow table" are used interchangeably even though there is a significant difference between a "cache" and a "table". Applicant respectfully disagrees with the Examiner that the use of these terms is confusing and inconsistent. Applicant has specifically indicated that in a preferred embodiment of Applicant's invention, the "shadow cache" is implemented as a temporary database table (Applicant's specification, paragraph [0048]). This is, for example, specifically indicated at paragraph [0049] of Applicant's specification which states as follows:

> In the currently preferred embodiment, the shadow cache is implemented by taking advantage of a temporary database that is used in conjunction with a SQL query engine to create a simple shadow table. The shadow table that is created is a very simple shadow table which contains a mapping of the block from the read-only cache view into a block that is to be used as a backing store in the shadow table. The shadow table is a very narrow table and it can be accessed very efficiently. In the presently preferred embodiment of the present invention the shadow table is keyed on block number and it also records a secondary block, which is the block that it maps to a block allocated in the temporary database.
>
> (Applicant's specification, paragraph [0049], emphasis added)

Applicant's claims also indicate that the shadow cache is implemented via a temporary database table. This is, for example, indicated in Applicant's claim 3 which includes the following claim limitations:

> The method of claim 1, wherein the shadow cache is implemented via a

temporary database table.

(Applicant's claim 3)

The term "shadow cache" is clearly described in the Applicant's specification and claims as being implemented as a temporary database table. Applicant has, however, made a correction to paragraph [0099] to use the same term shadow "table" consistently throughout the paragraph. Accordingly, Applicant believes that the Examiner's objection is overcome.

D. Objections to Claims

The Examiner objected to claims 2 and 18 as the limitation "to disk" lacks antecedent basis in the specification. Applicant has amended claims 2 and 18, thereby overcoming the objection.

The Examiner objected to Claims 4 and 20 are objected as including an informality and has also objected to these claims under 37 CFR 1.75(c), as being of improper dependent form. Applicant has canceled claims 4 and 20.

The Examiner objected to claims 8 and 24 as lacking antecedent basis in the specification. Applicant has amended claims 8 and 24, thereby overcoming the objection.

The Examiner objected to claims 9, 12, 25, and 28 as lacking antecedent basis in the specification. Applicant has canceled claims 9 and 25 and had amended claims 12 and 28, thereby overcoming the objection.

The Examiner has also objected Claims 13 and 29 as including a term not defined in the specification. Applicant has amended claims 13 and 29, thereby overcoming the objection.

E. Section 112, first paragraph rejection

The Examiner has objected to claims 1, 8, 9, 12, 13, 17, 24, 25, 28, and 29 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement. As per claims 1 and 17, the Examiner states that "cache view" limitation of these claims is not enabled because there is no indication of how it is implemented in the

software/hardware, i.e., as a cache, a table, a database, etc. Applicant respectfully disagrees with the Examiner that the "cache view" limitation is not enabled. Applicant's specification provides that the cache is used to create a view of the database a particular point in time. For example, paragraph [0047] of Applicant's specification provides as follows:

> With read-only transactions the methodology of the present invention provides for introducing another cache view. This new cache view is for the same database as the write cache view but it is a separate view of the database. In general terms, it is a view of a version of a database at a particular point in time. The approach of the present invention involves using the cache in order to create a view of the database at a particular point in time. Since details regarding all changes to the database are recorded in the transaction log file(s), the transactional log file(s) may be used to reconstruct a view of the database at this particular point in time. The view is constructed by applying log records to this view in the cache, so as to create a version of the database at a particular point in time.
>
> (Applicant's specification, paragraph [0047], emphasis added)

Although Applicant believes the term "cache" is well understood by those skilled in the art, Applicant's specification also includes specific description of this term. At paragraph [0061] of Applicant's specification, for example, the term "cache" is described as memory that contains recently accessed data (e.g., frequently-used pages of data from the database) and which is designed to speed up subsequent access to the same data.

The Examiner has also rejected claims 8 and 24 and claims 9, 12, 25 and 28 as not being enabled. As noted above in the discussion of the Examiner's objection to these claims, Applicant has amended claims 8, 24, 12 and 28 and has cancelled claims 9 and 25.

In addition, the Examiner also rejected the "posting back link log records" limitations of claims 13 and 29 as not being enabled. Applicant has amended claims 13 and 29 to state that back link log records are added to the transaction log. These limitations are specifically described in Applicant's specification, including, for example, at paragraph [0093] of Applicant's specification.

In view of the above remarks and the amendments to the claims made herein, Applicant respectfully suggests that the rejection of the above-referenced claims under 35

U.S.C. 112, first paragraph is overcome.

F. Section 112, second paragraph rejection

The Examiner has rejected claims 5-8, 11, 12, 21-24, 27, 28 and 30 under 35 U.S.C. Section 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which Applicant regards as the invention. Applicant has amended claims 5-8, 11, 12, 21-24, 27, 28 and 30. Accordingly, it is respectfully submitted that in view of the above-mentioned amendments, the rejection of Applicant's claims 5-8, 11, 12, 21-24, 27, 28 and 30 under Section 112, second paragraph as being indefinite is overcome.

G. Section 101 rejection

Claims 1-30 stand rejected under 35 U.S.C. 101 on the basis of non-statutory subject matter. Here, the Examiner states that Applicant's claimed invention does not appear to produce any useful result. Applicant has amended independent claims 1 and 17 to include claim limitations of returning a result comprising a transactionally consistent view of the given database supporting read-only uses. Accordingly, it is respectfully submitted that in view of the above-mentioned amendments, the rejection of Applicant's claims 1-30 under Section 101 on the basis of non-statutory subject matter is overcome.

Prior art rejections

A. Section 102(b): Hayashi

Claims 1-7, 9, 10, 12, 13, 15-23, 25, 26, 28, and 29 stand rejected under 35 U.S.C. 102(b) as being anticipated by Hayashi et al., U.S. Pat. 5,715,447 ("Hayashi"). For the reasons stated below, Applicant's invention may be distinguished on a variety of grounds.

Hayashi describes an approach that is fundamentally different from that of Applicant's. Hayashi's invention is directed to a technique for accelerating throughput, which is largely applicable for write transactions during OLTP (online transaction processing, such as an online reservation system where individual records are retrieved at random). Applicant's invention, on the other hand, keeps read-only transactions from being blocked by exclusive transactional locks of write transactions and also from having

long duration read transaction block write transactions with long duration shared transactional locks. Applicant's approach is applicable for DSS (decision support system) uses, such as reporting, where records are sequentially retrieved in large blocks. Hayashi's patent makes specific mention to the fact that his invention optimizes buffer locks, not transaction locks: "Locking is not a transaction lock but is to prevent a simultaneous execution of a buffer control function of an access module for accessing the database and a write back function of the contents of the object buffer to the disk (i.e., database)."

An important aspect of Applicant's invention is the use of "logical undo's" for reconstructing a transactionally consistent (i.e., proper, as of a given point in time) version of the database. In order to understand this aspect better, it is helpful to review undo operations. Physical undo and redo operations can be achieved by simply replaying the database log back and forth. A logical undo operation, however, must happen at a higher level -- that is, at the basic top level that records are added or dropped. Physical level, on the other hand, is more concerned about moving bytes, for example in the same manner that edits would occur in a text editor. In other words, in a physical undo or redo operation, the system need only move bytes around, for example undoing a change by copying prior contents (i.e., restoring a record by copying a prior byte sequence back into the record). And certainly in situations where possible, physical undo and physical redo operations provide an easy (if not the easiest) approach for undoing or redoing changes. In a multi-user database environment, it is often not possible to simply use physical undo and redo however.

Consider two database users: User A and User B. Although both users may not often be modifying the same database record, they will often be modifying the same database table, say an ORDER ENTRY table. Now, consider a scenario where a read-only transaction has begun (e.g., for generating a report or performing a data mining operation) and User A's changes to the ORDER ENTRY table have committed, but they are unfortunately intermingled with B's changes that have yet to commit. Since the database log (i.e., data structure tracking the foregoing changes) has recorded physical changes to the database table in sequential order, one cannot simply undo B's changes without undoing A's changes. In other words, one cannot physically undo B's changes in

a manner that will not affect A's changes.

In this scenario, the log also includes "logical" log records. These are high-level records that record logical (high-level) database operations, that is, ones that may be viewed as occurring at a top level, such as adding or dropping a database record. Note in particular that a high-level logical change to the table (e.g., adding a record) typically entails many physical changes besides the physical changes to the table itself (e.g., adding a byte sequence representing record data). For example, as a result of adding a record to a table any index on that table will likely need to be updated, thus requiring a physical change to affected indexes. Given the potential for a multitude of propagating changes or dependencies that may occur as a result of a single high-level change to a table, it is therefore helpful to represent these changes as high-level logical operations. In other words, a single high-level logical operation may be used to conveniently represent multiple lower-level physical operations.

The notion of a "logical undo" provides, in a similar manner, a higher-level representation of an undo (e.g., undo the insertion of a new record). This provides a higher-level representation than the multitude of physical undo operations that would need to be tracked to correctly represent the physical undoing of the operation against the entire database (i.e., physical undo of byte sequence change at the table, physical undo of the byte sequence change at the index, and so forth and so on).

Applicant's claims are not directed to just physical undo and redo operations, but instead are directed at a higher level: logical undo operations. As described above, in a given database environment, one may have multiple database users with intermingled write operations occurring against the database, some of which have committed while others are still pending. The notion of logical undo supports Applicant's approach of getting the database to a transactionally-consistent prior state that is suitable for performing long-duration read-only transactions (e.g., DSS, reporting, data mining, or the like). Importantly, Applicant's logical undo approach provides a mechanism to separate the ones that have committed from those that have not.

In accordance with Applicant's invention, a read-only transactionally consistent version is reconstructed by **logically undoing all incomplete transactions** at that point in time (i.e., at the point in time when the read-only version or view is to be

reconstructed). Here, transactions that were started but unfinished (at the time that the read-only transactionally consistent view is to be reconstructed) are logically undone. As a practical matter, all of these transactions include intermingled changes and therefore cannot simply be physically undone. If they could be physically undone, then the simplest approach would be to just proceed with physically undoing them. As a practical matter, they cannot be physically undone due to a multitude of intermingled dependencies.

Further, it is worth noting that logically undoing pending transactions is itself not a trivial undertaking. Apart from generating a number of potential edits to the shared cache view, the undertaking may substantially expand the shared cache (such as, for example, the logical undoing of a transaction inserting 1000 records). Had one been able to employ physical undo and redo, changes to the cache could have simply been applied on a per block basis. Logical undo cannot be handled on a per block basis, as the logical undo may in fact affect multiple blocks. To handle the possibility of the shared cache overflowing, therefore, a shadow database is therefore provided in accordance with Applicant's invention for receiving any overflow. This shadow database serves as a "backing store" for overflow blocks from the shared cache, thereby allowing the approach to avoid the computationally expensive task of rebuilding the cache once it has overflowed.

To be sure, shadow tables and databases have been used for lots of things, and Applicant does not claim to have invented them. Instead, Applicant's invention takes advantage of the technique to handle overflow that may occur in the above context. The shadow cache or database is only used if necessary as an overflow mechanism for reconstructed blocks. Traversing log files to reconstruct older versions of the database can be costly in terms of the log file reads required. Once a given block is reconstructed it is kept in the in memory cache. However, if the cache overflows, these reconstructed blocks can be written to a separate volatile temporary shadow database in Applicant's system. Here, the "shadow database" refers to a very specific cache used to support read-only transactions. There is no guarantee of durability for the shadow database because its contents are only needed for the duration of the read-only transaction(s). It does not serve as a shadow or backup database for the database itself.

Hayashi's approach, which is particularly geared to optimizing write operations, is fundamentally different from Applicant's approach. In particular, Hayashi (which appears focused on OLTP) essentially describes an optimization for write transactions, specifically "check pointing." Hayashi describes optimization techniques for reading and writing share blocks, especially the optimization of buffer locks used during nonvolitization. Hayashi describes the making of "copies" of buffers updated by write transactions during their process of nonvolitization. The Hayashi buffer "copies" do not constitute a transactionally consistent view of the database from a particular point in time.

Much of Hayashi's discussion focuses on the difference between "transaction locks" and their buffer locks. Applicant's invention, in contrast, optimizes read-only long duration transactions, by creating a transactionally consistent view of an entire database for read-only uses (e.g., decision support system (DSS) use, reports, data mining, or the like). Applicant's invention completely eliminates the need for "transaction locks". Instead, the invention uses log records to reconstruct a transactionally consistent, cached view of the database. As blocks are read into a read-only cache view they are made action (not transaction) consistent relative to when the read-only transaction began. A block is only made action consistent if it is accessed. By first reading the LSN (log sequence number) of an accessed block, Applicant's system can quickly detect if a block even needs to be made action consistent. The logical undo is performed on all incomplete write transactions at the start of a read-only transaction. Any blocks accessed after the logical undo operations will either can be used as is because their LSN is up to date or they will only need physical undo or redo operations which can be performed on the fly on a per block basis. The invention only makes "copies" of buffers if needed for a "read only transaction". This is a postmortem reconstruction of an older state using log records. All told, Applicant's approach and Hayashi's approach can almost be described as being virtual opposites of one another.

For the reasons stated, it is respectfully submitted that the claims set forth a patentable advance over the arts. In view of the remarks above (as well as clarifying amendments to the claims), it is believed that any rejection under Section 102 is overcome.

B. Section 103: Hayashi and Raz

Claims 8, 14, 24, and 30 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hayashi (above), in view of Raz, U.S. Pat. 5,701,480. Here, the Examiner adds Raz for the teaching of "database blocks that are computationally expensive to recreate." The claims are believed to be allowable for at least the reasons stated above pertaining to Hayashi. Specifically, Hayashi fails to teach or suggest the claim limitation set forth in Applicant's base claims of creating a transactionally consistent view of an entire database for read-only uses. Raz provides nothing to cure the deficiencies of Hayashi.

Any dependent claims not explicitly discussed are believed to be allowable by virtue of dependency from Applicant's independent claims, as discussed in detail above.

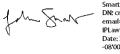
Conclusion

In view of the foregoing remarks and the amendment to the claims, it is believed that all claims are now in condition for allowance. Hence, it is respectfully requested that the application be passed to issue at an early date.

If for any reason the Examiner feels that a telephone conference would in any way expedite prosecution of the subject application, the Examiner is invited to telephone the undersigned at 408 884 1507.

Respectfully submitted,

Date: January 3, 2007



John A. Smart; Reg. No. 34,929
Attorney of Record

408 884 1507
815 572 8299 FAX